8P

УДК 004.4:519.85

К.А. Ручкин, Л.В. Холодов, Е.Г. Мухин
Государственный университет информатики и искусственного интеллекта,
г. Донецк, Украина

Распределённая разработка и анализ функциональности компьютерной системы для решения задач хаотической динамики

В статье продолжены исследования, начатые в [1] и посвященные организации процесса коллективной разработки компьютерной системы для решения задач хаотической динамики. Подробно описаны требования к функциональности компьютерной системы. Обоснован выбор архитектуры и методологии для дальнейшей разработки. Проанализированы средства для распределения математических вычислений. Подробно описан процесс разработки модуля визуализации.

Теория хаотической динамики является современным математическим аппаратом, методы и алгоритмы которого существенно помогают при решении многих научно-технических задач математики, механики, информатики, физики, химии, экономики, социологии и др. Например, таких как, прогнозирование погодных условий, прогнозирование поведения группы роботов и т.д. Однако существует, практическая сложность при исследовании динамических систем, описывающих реальные процессы, которая связана разработкой и реализацией компьютерно-аналитических методов. Создание универсальной компьютерной системы является важной и актуальной задачей, интересной не только математикам, но специалистам в области IT. В данной работе рассмотрены вопросы, посвященные организации процесса коллективной разработки компьютерной системы, предназначенной для решения задач хаотической динамики.

Постановка задачи

Целью исследования является разработка программного комплекса, применимого для решения определённого класса научно-технических задач, которые с математической точки зрения относятся к задачам хаотической динамики. Это накладывает на систему определённые требования.

1. Проведение большого числа вычислительных операций.
2. Настраиваемость под задачи пользователя (возможность решения различных задач класса).
3. Возможность работы как в режиме реального времени (on-line), так и в режиме off-line.

С технической точки зрения система должна удовлетворять следующим критериям качества программного обеспечения: производительность, гибкость, масштабируемость, предметно-ориентированность, интероперабельность.

С организационной точки зрения разработка данного программного комплекса связана с необходимостью использования методов коллективной разработки и потому требует разделения задач и выработки общих архитектурных решений. Это приводит к необходимости анализа современных технологий коллективной разработки сложных программных комплексов.

Система в целом состоит из набора отдельно разрабатываемых модулей, таких как вычислительный модуль, модуль визуализации, интерфейсный модуль и другие. В данной работе описывается разработка двух основных модулей системы – вычислительного модуля и модуля визуализации.

Основное назначение вычислительного модуля состоит в решении задач хаотической динамики с применением распределённых вычислительных средств. Основное назначение модуля визуализации – графическое представление результатов решения задач хаотической динамики в реальном времени.

Выбор средств распределения вычислений для вычислительного модуля

Задачи хаотической динамики отличаются значительной вычислительной сложностью, в связи с этим представляется актуальным проведение исследования способа распараллеливания проводимых вычислений на компьютерах, соединённых сетью. Так как вычислительные клиенты должны работать на соединённых сетью машинах, за которыми работают пользователи, к ним выдвигаются следующие требования:
- работа в системе Windows;
- простота установки и настройки;
- возможность отключения в ходе работы;
- незаметная для пользователя ПК работа в фоновом режиме.

Во многих существующих публикациях о разработке параллельных программ обращается внимание на стандарт MPI, который сегодня является стандартом де-факто для реализации параллельных вычислений в системах с распределённой памятью и используется во многих суперкомпьютерных задачах. Поэтому было решено исследовать применимость именно этого стандарта.

MPI – это программный интерфейс (API) и его реализация, позволяющий обмениваться сообщениями между компьютерами, выполняющими одну задачу. В настоящее время MPI является наиболее распространённым стандартом интерфейса обмена данными в параллельном программировании, существует его реализация для большого числа компьютерных платформ, в том числе и для Microsoft Windows. Стандарт MPI фиксирует интерфейс, который должен соблюдаться как системой программирования на каждой вычислительной платформе, так и пользователем при создании программ, таким образом, MPI-программы являются переносимыми и для их запуска в гетерогенной сети достаточно провести компиляцию под каждую необходимую платформу, кроме того это гарантирует, что для переноса приложения с одной реализации MPI на другую необходимо только провести пересборку, без какого-либо изменения исходного кода [2].

Основным средством коммуникации между процессами в MPI является передача сообщений друг другу. Поддерживается блокирующая и неблокирующая пересылка сообщений. MPI поддерживает работу с языками программирования Fortran, C и C++. Интерфейс MPI поддерживает создание программ по принципу MIMD (Multiple Instruction Multiple Data).

В качестве возможного средства реализации распределённых вычислений были рассмотрены доступные реализации интерфейса MPI, в которых декларируется поддержка последней версии стандарта MPI-2.0 и совместимость с Windows платформой – это MPICH и WMPI. Для данных реализаций также декларируется поддержка работы с распределением задач между компьютерами в сети, в том числе в гетерогенных сетях.

С целью исследования возможности применения MPI было разработано тестовое приложение с использованием библиотеки MPICH, реализующее параллельное вычисление одной из требуемых вычислительных задач. При запуске разработанного

теста с двумя одновременно работающими экземплярами программы на процессоре Pentium с технологией Hyper-Threading скорость вычислений возросла на 62 %, что говорит об очень хорошей эффективности распараллеливания с помощью MPICH на одном компьютере. При тестировании распараллеливания вычислений между двумя компьютерами, соединёнными локальной сетью, скорость вычислений выросла на 93 %.

Для библиотеки WMPI производителем декларируется до 40 % большая производительность по сравнению с другими реализациями MPI. Тестирование разработанного приложения, собранного с использованием библиотеки WMPI, действительно показало немного большую эффективность распараллеливания (2 – 3%) в сравнении с MPICH. Однако в системе Windows Vista приложения, собранные с данной библиотекой, выдают сообщение об устаревшей версии Windows, при запуске же в режиме совместимости с WindowsXP или более ранними приложениями аварийно завершаются. К сожалению, эта библиотека с 2006 года не обновлялась.

Сводная диаграмма эффективности распараллеливания с использованием различных библиотек приведена на рис. 1.

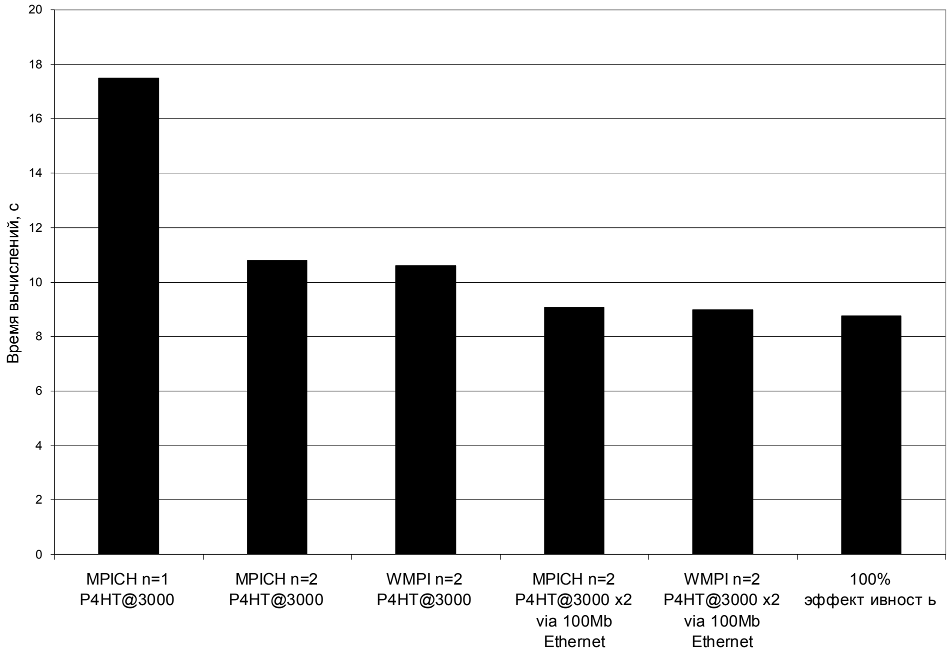


Рисунок 1 – Сводная диаграмма эффективности распараллеливания построения огибающей поверхности

Таким образом, единственной применимой в Windows среде с бесплатной реализацией MPI является MPICH, однако при проведении вычислений на нескольких хостах был обнаружен ряд её недостатков:

– на каждом хосте необходимо наличие учётной записи пользователя с одинаковыми именем и паролем;

– на каждом хосте необходимо наличие исполняемого файла, а в случае необходимости и всех необходимых файлов, к которым происходит обращение, так как MPI не производит копирования при запуске;

– каждый из исполняемых файлов должен быть разблокирован в брандмауэре;

– отсутствует встроенный механизм обнаружения разрыва связи между хостами, при разрыве связи возможен вход в состояние бесконечного ожидания завершения операции обмена сообщениями;

– отсутствует встроенный механизм установки приоритета приложения в системе.

Проведённое исследование показало, что в контексте данной задачи применение MPI целесообразно в том случае, если вычисления будут проводиться на выделенном вычислительном ресурсе. В случае использования MPI на пользовательских машинах, развёртывание и эксплуатация системы затруднены.

В данный момент рассматриваются следующие альтернативы: Condor, Globus, X-Com, UNICORE.

Особенности разработки модуля визуализации

В процессе разработки системы компьютерного моделирования и анализа задач хаотической динамики возникла необходимость реализации модуля визуализации, удовлетворяющего следующим требованиям: быстрота визуализации большого количества точек траекторий, независимость от аппаратных средств, высокая детализация получаемых изображений, настраиваемость модуля в соответствии с возможностями аппаратуры пользовательской платформы.

Исходя из указанных выше требований, была поставлена цель исследования: разработать модуль визуализации для системы компьютерного моделирования задач хаотической динамики, абстрагирующий функциональность современных низкоуровневых аппаратно-программных средств вывода трёхмерной геометрии и не имеющий от них прямых зависимостей.

Дополнительное требование отсутствия прямых зависимостей модуля становится актуальным в связи с большой сложностью современных аппаратно-программных интерфейсов, таких как OpenGL и DirectX, их высокой изменчивостью, а также несоответствием некоторых аппаратных средств спецификациям, предъявляемым вышеуказанными API.

Рассмотрев поставленную задачу с точки зрения архитектуры программных систем, было принято решение разделить модуль на 3 компонента: компонент визуализации трёхмерных сцен, компонент визуализации трёхмерных примитивов и компонент низкоуровневой визуализации. Таким образом, в качестве базовой архитектуры для модуля была выбрана архитектура слоев. Инкапсуляция низкоуровневых особенностей визуализации трёхмерной геометрии осуществлена за счёт применения принципа инверсии зависимостей DIP. Общая схема компонентов модуля приведена на рис. 2.

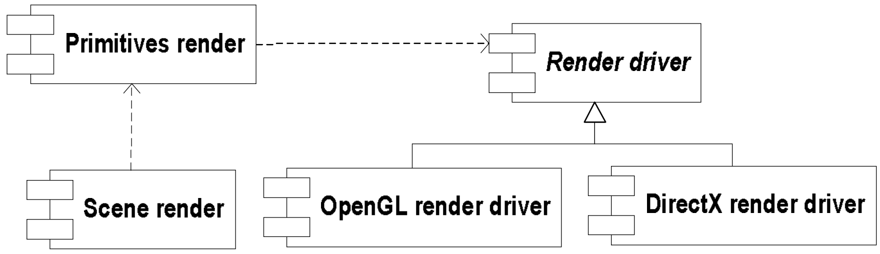


Рисунок 2 – Общая схема компонентов модуля визуализации

Применение принципа инверсии зависимостей для организации компонентов позволяет решать две различные задачи независимо друг от друга. Так, особенности реализации модуля на OpenGL не влияют на алгоритмы визуализации компонента Primitives render, тем самым упрощая разработку модуля. Помимо этого возникает потенциальная возможность реализации модуля визуализации с поддержкой DirectX 10 в случае, если возможностей OpenGL будет недостаточно.

С точки зрения реализации проектного решения наиболее сложным является компонент Render driver, который должен, с одной стороны, обобщить функциональность современных графических API, а с другой – предоставить возможность эффективной реализации вынесенной в интерфейс функциональности с учётом особенностей аппаратной платформы пользователя.

Для решения проблемы обобщения графических API было принято решение использовать паттерны объектно-ориентированного проектирования, в частности Abstract Factory и Strategy. Совместное применение этих паттернов позволяет не только упростить задачу, но и позволяет изменять поведение всех компонентов, входящих в модуль, без модификации исходного кода, что соответствует принципу OCP разработки программного обеспечения.

В результате применения вышеуказанных паттернов был спроектирован интерфейс компонента RenderDriver, диаграмма статического представления которого приведена на рис. 3.

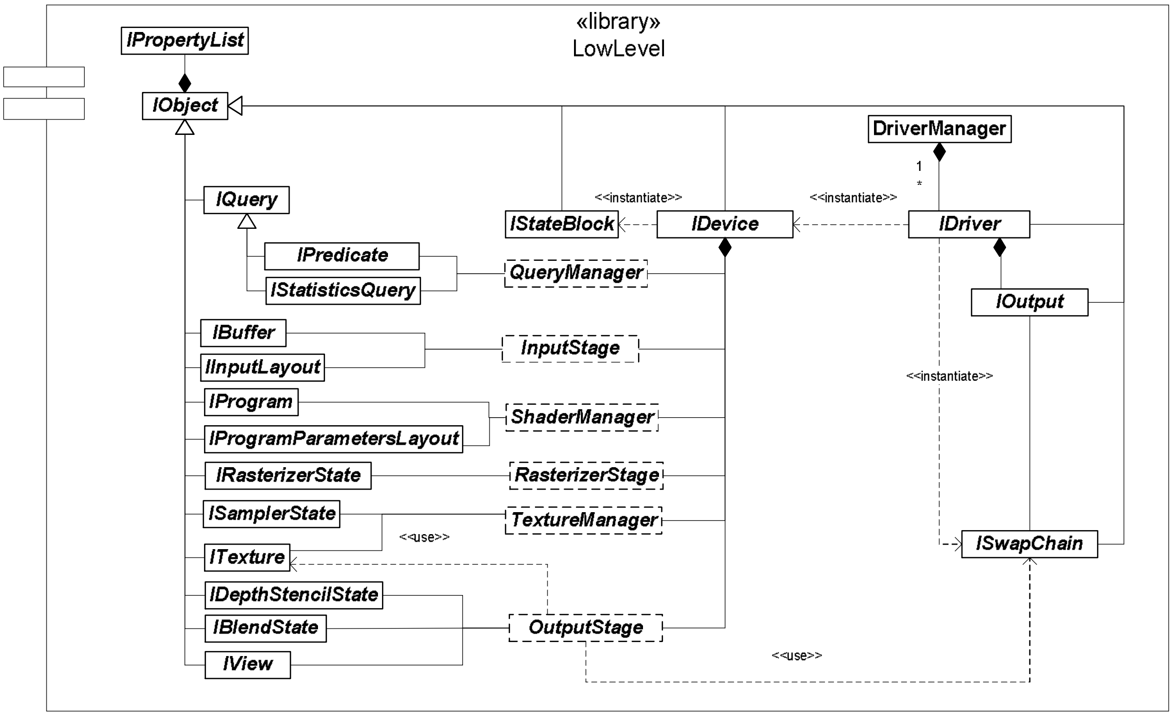


Рисунок 3 – Диаграмма статического представления интерфейса компонента RenderDriver

В процессе проектирования компонента RenderDriver были выделены следующие интерфейсы:

– IBuffer – интерфейс работы с буферами геометрических данных (положение точек, их цвет, нормали);

– ITexture – интерфейс работы с текстурами (загрузка и получение битовых образов различного формата);

– ISamplerState, IBlendState, IDepthStencilState – интерфейсы объектов состояния устройства визуализации (параметры смешивания цветов, настройки буфера отсечения и т.п.);

– IDevice – интерфейс устройства визуализации (вывод на экран трёхмерных примитивов, задание настроек визуализации).

Основные проблемы при проектировании модуля заключались в необходимости обобщения функциональных возможностей низкоуровневых графических API с возможностью эффективной реализации, учитывающей аппаратные возможности компьютера пользователя. При проектировании обобщались возможности таких API, как DirectX 10 и OpenGL 2.1 – двух основных применяемых на платформе PC. При обобщении был выявлен ряд противоречий, среди которых – невозможность обобщить различия в отдельных аппаратных конфигурациях, а также сложность и громоздкость обобщения отдельных мест OpenGL и Direct X.

Для решения указанных проблем были применены следующие методы проектирования.

1. Уровневое разбиение функциональности графических API и определение связей между уровнями.

2. Применение паттернов объектно-ориентированного проектирования для изоляции изменчивых мест в программном коде.

3. Выбор в качестве эталона интерфейса DirectX 10 и реализация его на базе OpenGL, что позволило постоянно согласовывать два API, учитывая их особенности реализации, и получить проектное решение, пригодное для реализации как на базе OpenGL, так и на базе DirectX.

4. Применение метода проектирования по контракту [4], заключающегося в обязательной проверке во всех методах при реализации предусловий и инвариантов классов.

Одним из основных уровней в модуле визуализации является модуль наложения текстур. Данный подуровень скрывает особенности реализации различных видов текстур, а также эмулирует отсутствие отдельных расширений OpenGL для работы с текстурами. В частности, эмулируются расширения GL_ARB_texture_non_power_of_two, позволяющего работать с текстурами, размеры которых не являются степенью двойки.

Взаимодействие спроектированных интерфейсов с классами реализации на OpenGL показано на рис. 4.

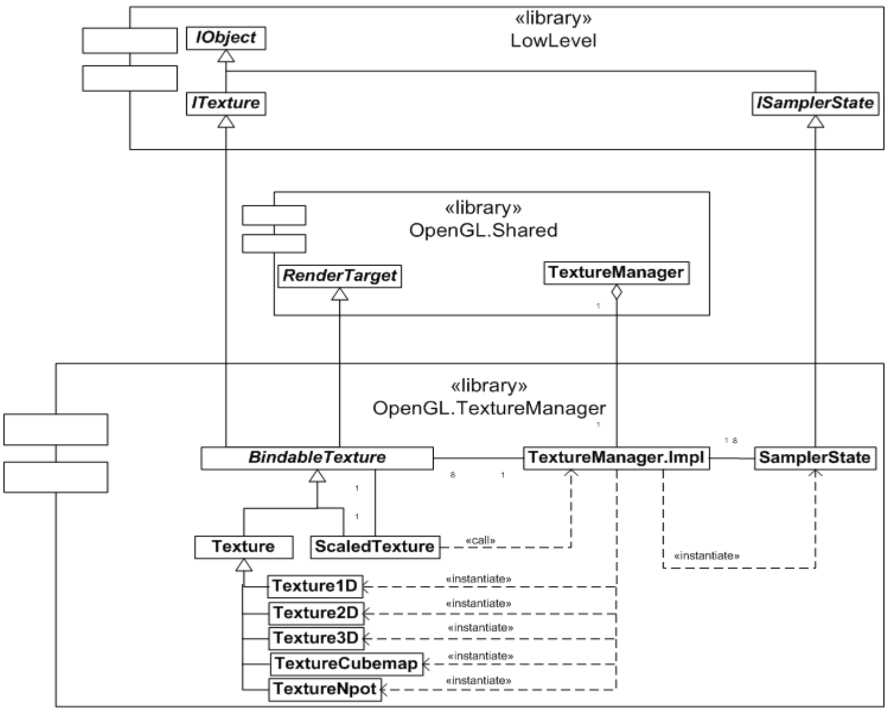


Рисунок 4 – Диаграмма статического представления компонента OpenGLDriver

На диаграмме представлены три уровня.

1. Интерфейсный уровень LowLevel компонента RenderDriver.
2. Промежуточный уровень OpenGL.Shared, обеспечивающий взаимодействие уровней реализации компонента OpenGLDriver.
3. Уровень OpenGL.TextureManager, отвечающий за наложение текстур.

Существенным при реализации уровня OpenGL.TextureManager является сосредоточение средств работы с текстурами в рамках одной и только одной библиотеки, а также замыкание создания объектов текстур в реализации класса TextureManager.Impl.

Сравнивая количество классов компонента OpenGLDriver и интерфейсного по отношению к нему компонента LowLevel, можно говорить о значительном снижении сложности при работе с компонентом по сравнению с работой непосредственно через OpenGL.

Разделение, подобное описанному выше, было произведено в рамках реализации всего компонента OpenGLDriver. Это позволило выделить группы совместно модифицируемых классов и инкапсулировать их, что в свою очередь снизило общую сложность системы.

Таким образом, в рамках разработки системы компьютерного моделирования и анализа задач хаотической динамики был спроектирован модуль визуализации. Модуль был разбит на три компонента, один из которых – компонент низкоуровневой визуализации – был реализован на базе библиотеки OpenGL.

С технической точки зрения модуль содержит 75 основных классов, обеспечивающих функционирование порядка 65 функциональных точек. Общий объём кода модуля на C++ – 16 000 строк кода. На сегодняшний день модуль поддерживает OpenGL версий 1.1 – 2.1 и порядка 30 его расширений. Отсутствие ряда расширений на машине пользователя эмулируются средствами модуля.

Выводы

Таким образом, в рамках разработки системы компьютерного моделирования задач хаотической динамики были решены следующие задачи.

1. Организация процесса разработки.
2. Анализ инструментальных средств распределения вычислений, необходимых для реализации модуля визуализации.
3. Проектирование модуля визуализации.
4. Реализация низкоуровневой части модуля визуализации на базе библиотеки OpenGL.

Литература

1. Ручкин К.А., Холодов Л.В. Распределенная разработка программного обеспечения системы компьютерного моделирования задач хаотической динамики // Искусственный интеллект. – 2008. – № 2. – С. 41-43.
2. Гецци К., Джазайери М., Мандриоли Д. Основы инженерии программного обеспечения. 2-е изд.: Пер. с англ. – СПб.: БХВ-Петербург, 2005. – С. 640-647.
3. Мартин Р., Ньюкирк К., Косс Р. Быстрая разработка программ: принципы, примеры, практика. – М.: Издательский дом «Вильямс», 2004. – 752 с.
4. Meyer B. Design by Contract, in Advances in Object-Oriented Software Engineering. – Prentice Hall, 1991. – P. 1-50.
5. Библиотека WMPI. – Режим доступа: http://www.criticalsoftware.com/wmpi.html.
6. Антонов А.С. Параллельное программирование с использованием технологии MPI. – М.: Изд-во МГУ, 2004. – 71 с.
7. Фостер Ян. Designing and Building Parallel Programs – Режим доступа: http://www-unix.mcs.anl.gov/dbpp/
8. Приёмы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. – СПб.: Питер, 2004. – 366 с.
9. Библиотека MPICH2. – Режим доступа: http://www.mcs.anl.gov/research/projects/mpich2/.
10. Библиотека OpenGL – Режим доступа: http://www.opengl.org.

К.А. Ручкін, Л.В. Холодов, Є.Г. Мухін

Аналіз методів розподіленої розробки комп'ютерної системи для розв'язання задач хаотичної динаміки

У статті запропоновані дослідження, які розпочаті [1] та присвячені організації процесу колективної розробки комп'ютерної системи для розв'язання задач хаотичної динаміки. Детально описані вимоги щодо функціональності комп'ютерної системи. Обґрунтований вибір архітектури і методології щодо подальшої розробки. Проаналізовані засоби для розподілу математичних обчислень. Детально описаний процес розробки модуля візуалізації.

Статья поступила в редакцию 09.07.2008.